Paying Attention to the Insider Threat

Eduardo Lopez
Information Systems, DeGroote School of Business
McMaster University
Hamilton, ON, Canada
lopeze1@mcmaster.ca

Kamran Sartipi
Department of Computer Science
East Carolina University
Greenville, NC, USA
sartipik16@ecu.edu

***Abstract*—The misuse of information systems by internal actors – the insider threat – is an ever-growing concern in organizations of all types. The timely detection of an insider threat is as important as it is difficult. Analyzing user behaviors recorded in electronic logs require significant computing resources and the capability to find and interpret complex patterns in temporal sequences that may contain irrelevant, temporary or novel elements. In this paper we use an attention-based architecture derived from BERT (Bidirectional Encoder Representations from Transformers) for the creation, storage and updating of an always-current, holistic user behavior model that enables real-time insider threat detection through anomaly detection and user behavior prediction techniques. A case study with a very large transaction system is provided.**

Keywords: insider threat, transformers, BERT, anomaly detection, cybersecurity[1]

I. Introduction

Detecting an on-going insider threat is a significant challenge. Although the actions by every user are regularly recorded in electronic files (i.e., logs), those logs can be obscured by a very large number of unrelated events. Information in electronic logs is usually unstructured, stored in very large text files that require specialized tools to be analyzed. User behaviors are captured in sequences of events that can be mined for abnormal patterns. However, their ever-evolving characteristics and dependence on a variety of contextual parameters (e.g., the time of use or location of computer) pose remarkable challenges for advanced analytic applications. For the purpose of this work, we focus on a set of technologies known as *Transformers* [13] that have dramatically improved Artificial Intelligence (AI) applications in Natural Language Processing (NLP). They are encoder/decoder architectures that implement the concept of *Attention* [3].

The preceding decade has witnessed remarkable advances in Artificial Intelligence (AI). Deep learning in particular has consistently delivered results across myriad disciplines, sometimes surpassing human-level benchmarks [1]. A deep learning architecture is a multi-layer stack of neural network units where usually most of them are subject to learning and that may include non-linear input-to-output mappings [8]. Each layer in the network incrementally learns about the structure of data from its preceding layers, becoming very good function approximators that can find and learn very complex patterns in the data. There are several deep learning architectures that can be applied to specific applications and data types.

Deep learning has been proven effective across a wide range of disciplines, with cybersecurity being the focus of this work. The protection of information assets against malicious threats is a pivotal element for an increasingly technified society where information systems enable processes for many organizations. Arguably, one of the most interesting challenges pertains to the phenomena known as the insider threat. It can be defined as current (or former) users – or somebody impersonating them – that intentionally misuse access privileges negatively impacting the confidentiality, integrity or availability of information or information systems [4]. As an anomaly detection exercise, deep learning is an effective tool to find such patterns. User behavior is non-linear, complex and difficult to ascertain unless a strong function approximator such as a deep learning network can be used.

In this study we train a user behavior model as the baseline to perform anomaly detection and behavior prediction for insider threat detection. In particular, this paper contribures to the cybersecurity literature by proposing a Transformer-based architecture that uses self-attention for modeling user behaviors. The model created can be used effectively in transfer learning tasks.

This paper is organized as follows: Section II presents the relevant research to our work. Section III articulates the architecture of Transformers. Section IV presenst the proposed approach and Section V elaborates on the case study we conducted. Section VI draws conclusion to our discussion.

II. Related Work

The detection of the insider threat with machine learning can be formulated in multiple ways. Lopez and Sartipi [10] articulated the analysis as a supervised-learning classification, where the existence of a known pattern or signature is used by a logistic regression classifier to determine if the user behavior indicates information systems misuse. Liu et al [9] present an example for the use of autoencoders in insider threat detection. Paul and Mishra [12] propose an LSTM-based approach to threat detection. LSTM architecture' effectiveness in insider threat detection has also been demonstrated by Lopez and Sartipi [11], and by Paul and Mishra [12]. More advanced

[1]DOI reference number: 10.18293/SEKE2022-059

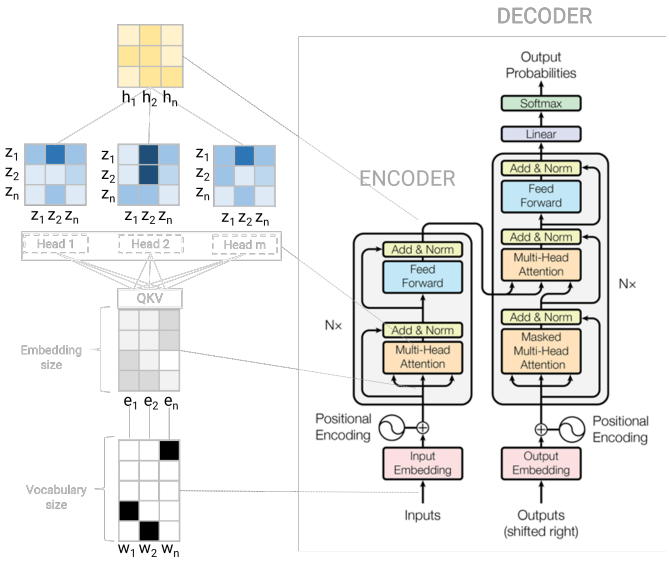

Fig. 1. Transformer architecture adapted from [13] .

architectures leveraging LSTM include stacked models, pre-initialized LSTM and bi-directional LSTM which have produced better results than single LSTM layers [6]. The concept of attention and in particular the Transformers proposed in the seminal work by Vaswani et al. [13]. From that moment on, the rate of new advances in Natural Language Processing has been unrelenting. Interestingly enough, the use of Transformers in cybersecurity applications has been rather limited. To our knowledge, no work has been done in applying these concepts to insider threat detection.

III. Transformer

The concept of *Attention* first appeared in 2015 [3] but it was until the Transformers concepts were proposed in 2017 that a new leap in performance was attained [13]. In its original form, a Transformer follows an encoder-decoder architecture as depicted in Figure 1. The original implementation uses six stacked encoders and six stacked decoders; however, since then multiple evolutions have taken place with different values and parameters. As it is the case with most deep learning architectures, the learning model uses successive representations of the original data, obtaining features at a higher abstraction level.

The input to a transformer model is usually raw data in the form of a sequence. Figure 1 represents this as a one-hot encoded table from words w_1 to w_n. The first step in the encoder is to transform the input into meaningful dense-vector representations. In contrast to recurrent neural networks, all the words in the sequence are fed at the same time to the encoder, so the word placement in the sequence is not explicitly captured. To add this information, a positional encoding takes place to calculate the words' positional embeddings. The encoder usually has multiple, parallel self-attention heads used during training. Each head can be considered a representation subspace where different attributes of the embeddings are calculated. In the case of NLP, a useful intuition is to consider each head as a representation of a different words' attributes, i.e., one may represent grammar and the other one may represent syntax or gender. The original Transformer uses eight heads with each one randomly initialized at training time.

Self-attention is the next step in the encoder which is considered fundamental to the concept of transformers. Reverting back to NLP for intuition: let us supposed we have the sentence *"Transformers are great for long sequences because **they** use attention"*. Any learning model needs to clearly determine if the word *they* refers to "Transformers" or to "long sequences". A way to formulate and quantify this dependency is by calculating for each word a vector to represent the contribution of itself and every other word in the sentence. Figure 1 depicts the self-attention as *m* square matrices of length *n*, where *m* is the number of attention heads in the architecture. Equation 1 shows the different operations performed.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad (1)$$

Q, *K* and *V* are three abstractions for Query, Key and Value, which are calculated by multiplying the word embeddings with weigh matrices W^Q, W^K and W^V learned during training. The dot product of *Q* and *K* is scaled down by the square root of the dimension of the *K* matrix. A softmax transformation is then applied (producing a matrix with values from 0 to 1) and then multiplied by the V matrix to produce the attention matrix.

There are multiple transformer designs based on the original work by Vaswani [13]. For reasons that will be explained in section IV, we proceed to elaborate on the architecture known as BERT. The term stands for Bi-directional Encoder Representation from Transformers [5], and it uses only the encoder part of the transformer architecture. A fundamental characteristic of BERT is that it learns the underlying patterns of the data considering both the left and right context of the sequences used. BERT uses two unsupervised learning objectives: a Masked Language Model (MLM) and Next Sentence Prediction (NSP). These two tasks are executed over a large text corpus during a stage known as *pre-training*. Once the parameters have been found, the model is ready for prediction or classification tasks during a *fine-tuning* stage. Pre-training BERT can be conceptualized as building the underlying knowledge about the dataset. This knowledge can be then applied to other tasks with a smaller dataset, i.e., an example of *transfer learning*. While pre-training may be a lengthy task, fine-tuning is very efficient. Pre-trained BERT models are publicly available for NLP tasks. BERT large uses 24 encoders stacked, representing each word with 1024 dimensions and 16 self-attention heads. The total number of parameters learned for BERT large is 340M.

IV. Approach

Dataset. We use the Los Alamos cybersecurity events dataset that is publicly available for research [7]. It includes multiple files from different information systems containing authentication events, programs started or finished, Domain

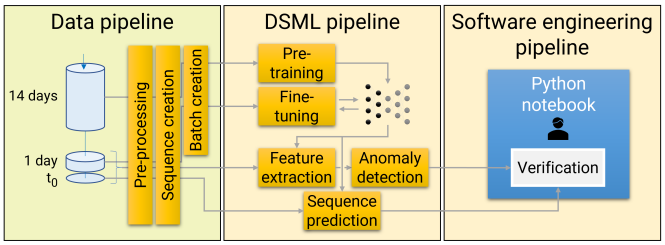

Fig. 2. Architecture for insider threat detection.

Name Service (DNS) calls and network flows. The dataset is the result of 58 days of continuous monitoring of more than 12,000 users and 17,000 computers. There are a total of 1.6 billion records contained in 17GB of data. In addition to this, there are 728 authentication records capturing the events from a Red Team, which are individuals purposely performing tasks that are typical of insider threats.

Architecture. To describe the architecture selected for the experiments, we use industry best practices, categorizing three groups of processes: i) Data engineering pipeline; ii) Data Science and Machine Learning (DSML) pipeline, and; iii) Software engineering pipeline [2]. In the data pipeline we transform the raw data into information that can be entered in the DSML model, with its output provided through a user interface so the user can verify the results.

We select BERT as the foundational architecture for the DSML pipeline. As was explained in Section III, BERT is an encoder that uses left and right contexts from sequences. Upon pre-training, BERT effectively creates a model (equivalent to a Masked Language Model in the original BERT implementation) that can be used in downstream tasks such as prediction or classification. In the scenario explored in this study, BERT produces a user behavior model that can be leveraged for the detection of the insider threat. In order to provide BERT with the required data, the data engineering pipeline pre-processes the data and crafts the sequences that will be used in the DSML model. Figure 2 depicts the complete architecture.

There are three distinct time windows used for the insider threat detection. The first one is the historical data used for pre-training. Based on domain knowledge, 14 days provides sufficient data to capture repetitive user patterns. Keeping the time window short, the pre-training time can be optimized so the learning process and can be completed in a timely fashion. Once the model has been pre-trained, a daily fine-tuning can take place. By using this strategy, the user behavior model is kept current with novel patterns performed by users. Fine tuning with 1-day data can be done overnight. The final time window used is one second. This means that the system will use the information collected within the log in real-time, and performs the analysis using the user behavior model.

The DSML component is where the user behavior model is estimated and stored. The pre-training task estimates the parameters of the deep learning model using the historical data. The fine-tuning adjusts the parameters based on the information from the preceding day. The current data (i.e., t_0) is then used in the detection of the threat. To achieve this objective, the architecture performs two sets of analysis: word prediction and anomaly detection.

V. Case Study

Data engineering pipeline. From the data tables available in the Los Alamos cybersecurity events dataset [7] we select *authentications* as the source for the analysis. Figure 3 shows the elements of an authentication record.

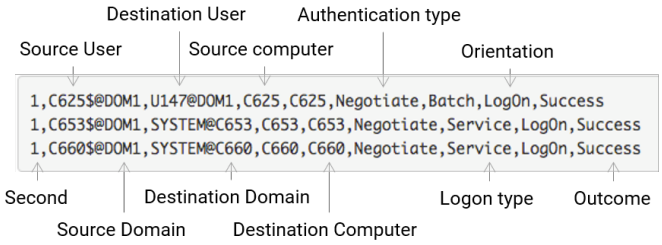

Fig. 3. Transforming authentications to sequences.

There are 1.051 billion authentication records in the dataset. Each record includes the time (in seconds) when it happened, as well as other key data such as the users, computers, type and direction involved in the authentication. As can be observed in Figure 3, there are multiple authentication events in any given time instance which involve different entities. Through the pre-processing stage we transform the dataset and obtain the 'words' to be used in downstream processes.

The next stage is the creation of sequences, first into 'sentences' and later on into 'documents' where sentences are separated by the special token [SEP]. To build the sentence structure we consider the most optimal, meaningful information that will enable the threat detection. Through experimentation we find that the maximum number of tokens that support short pre-training and fine-tuning cycles is 256. Conversely, a 'short' sentence may contain less information but enable the use of longer sequences and still succeed at detecting the insider threat. A review of the data shows that the user behavior on any given second (which in NLP is analogous to a document) may contain anywhere from 250 to 3,000 tokens (i.e., words), with the median around 2,000. We select 2048 as the maximum sequence length. Upon tokenization, we determined that the total number of unique events – the vocabulary size is around 30K. The data is now ready to be used by the DSML pipeline.

DSML pipeline As mentioned before, we adopted BERT as the deep learning architecture for the detection of the threat. The original BERT base design was composed of 12 layers (i.e., stacked transformer encoders) using 768 dimension for the hidden states and 12 attention heads. The total number of trainable parameters is 109M, equivalent to all the weights that will be adjusted during training. Given the time constraints that we have articulated, and the efficiency we require in the process to enable fast response, we reduce the model size by decreasing the number of heads and encoders to six and the hidden state size to 384. Although the vocabulary size is

Len	Sequence	Cumulative probability >50%	[MASK]	Ground truth
565	... hour10 weekday3 [MASK] C585 C585 ...	U2753	U2753	Normal
216	... hour4 weekday3 U8 [MASK] C423 ...	C568 C62	C62	Normal
85	... hour8 weekday4 U77 C616 [MASK] ...	C2742 C616	C616	Normal
693	... hour10 weekday1 U1506 [MASK] C2388 ...	C2388 C3610	C17693	Threat
397	... hour7 weekday1 U8946 [MASK] C19038 ...	C19038	C17693	Threat
389	... hour18 weekday2 U8840 C17693 [MASK] ...	C625 C612 C467 C586 C457	C370	Threat

Fig. 4. User behaviours prediction for insider threat detection.

comparable to that of the English language, the short sentence structure has just five unique types of entities: *computers, users, hours* and *weekdays*. In contrast, a natural language uses myriad different token types. This simpler architecture uses approximately 30M trainable parameters, driving contained and efficient resource usage while still representing each token by rich 384-dimensional vectors. We proceed with the pre-training of the model using 14-day historical data, or 1,209,600 documents (i.e., seconds). When using an NVIDIA V100 GPU and the short sentence structure the pre-training takes approximately 40 hours for 4 epochs. The fine tuning of the user behavior model is performed using the data from the preceding day. As expected, resuming the training with the latest daily data can be done quite rapidly (e.g., overnight). The user behavior model is now suitable for use by the core detection processes.

Insider threat detection. The user behaviour model learned in the pre-training stage can also be used for sequence prediction. A key advantage of a BERT-centric architecture is the ability to perform transfer learning. Any token in the sequence can be masked and predicted with the model. We select the current second (t_0) in Figure 2, pre-process the raw data and create the sequence to be inputted in the model. Figure 4 displays six different instances for analysis.

The first three sequences correspond to normal user behaviors captured in the data. The first sequence has 565 tokens, and it is entered into the model masking the user name. All predicted values whose probabilities add up to more than 50% are displayed. For the first sentence, only one value is needed for surpassing the 50% threshold; this is U2753 with a probability of 90.56%. The model is quite certain about the prediction, and the actual value is indeed U2753. Therefore, we can consider the behavior as normal.

The second sequence has 216 tokens in it, and we now mask the source computer. The two highest-probability predictions cumulatively reach the 50% threshold: C568 and C62. The actual source computer used by the user U8 was C62, so the model is once again correct with no indication of an insider threat taking place. The third sequence is very short, just 85 tokens, and we mask the destination computer. As it was the case in the preceding one, the correct computer C616 is predicted, so we can consider the behavior a normal one.

We now proceed to analyze three known *threat cases*. It is important to emphasize that this ground truth data is used to evaluate the accuracy of the prediction, but was not used when training the model. In the first case we mask the source computer for user U8946 in a 693-tokens sequence. Two computer predictions accumulate beyond 50%: C2388 and C3610, but none of them is the actual computer in the data, i.e., C17693. In this case, none of the predictions ended up being correct. This is an indicator of an insider threat that would be communicated to a human for further review. The incorrect prediction by the model is a correct indicator of an insider threat. The fifth sequence is 397-tokens long, and we mask again the source computer. The model in this case is also strongly convinced that the source computer must be C19038 (with a very definitive 99% probability). However, the actual source computer is different, which is a clear indicator of an abnormal event, as the model has high confidence in a prediction that ends up being incorrect. In the last case we mask the destination computer and enter the data in the model. Many predictions are needed to reach the 50% threshold – which can be interpreted as the model having difficulties to predict with a high-level of certainty. The actual value C370 is not in the prediction group, which again points to a potential insider threat and shall be sent to a human for review.

VI. Conclusion

This research formulates the insider threat detection as an attention-based machine learning problem. The objective is to effectively detect a threat taking place with optimal efficiency driving a timely response from a human actor. We demonstrate how a Transformer deep learning configuration based on the BERT architecture achieves this objective by leveraging the strengths of an attention-based configuration. We demonstrated how to identify a potential insider threat in near real-time using a well-defined three-step machine learning process.

References

[1] AI index report 2021. https://aiindex.stanford.edu/report/, 2021.
[2] S. Barot. Realizing Value From Composable AI Through XOps.
[3] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-Based Models for Speech Recognition. *arXiv:1506.07503 [cs, stat]*, June 2015.
[4] D. L. Costa, M. J. Albrethsen, M. L. Collins, S. J. Perl, G. J. Silowash, and D. L. Spooner. An Insider Threat Indicator Ontology. page 87.
[5] Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019.
[6] A. Graves, N. Jaitly, and A.-r. Mohamed. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Olomouc, Czech Republic, Dec. 2013. IEEE.
[7] A. D. Kent. Comprehensive, Multi-Source Cybersecurity Events, 2015.
[8] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
[9] Liu et al. Anomaly-Based Insider Threat Detection Using Deep Autoencoders. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 39–48, Nov. 2018.
[10] E. Lopez and K. Sartipi. Feature Engineering in Big Data for Detection of Information Systems Misuse. page 12, 2018.
[11] E. Lopez and K. Sartipi. Detecting the Insider Threat with Long Short Term Memory (LSTM) Neural Networks. *arXiv:2007.11956 [cs]*, July 2020.
[12] S. Paul and S. Mishra. LAC: LSTM AUTOENCODER with Community for Insider Threat Detection. In *2020 the 4th International Conference on Big Data Research (ICBDR'20)*, pages 71–77, Tokyo Japan, Nov. 2020. ACM.
[13] Vaswani et al. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.